SF3847 Convex optimization with engineering applications
Spring 2023
Instructors: Mats Bengtsson, Anders Forsgren and Joakim Jaldén

Homework Assignment 4
Due Tuesday May 9 2023

In this final homework, you will write your own algorithm for solving a given convex problem.

Since this problem is more extensive and time-consuming than those of the previous homeworks, the maximum number of points is 40, even though it's a single problem.

Exercise 4.1 a) Implement an algorithm (using Matlab, Python, or any other programming language of your choice) that solves the following non-linear programming problem. Your algorithm should be implemented from first principles, i.e. it may use standard linear algebra libraries (for example the standard matrix/vector operations and functions in Matlab/NumPy), but not any available optimization subroutines.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{i=1}^{m} \log(1 + x_i c_i) \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Assume that $c \in \mathbb{R}_+^m$, $A \in \mathbb{R}_+^{n \times m}$ and $b \in \mathbb{R}_{++}^n$. Show that these assumptions make it easy to find a feasible interior starting point. Choose yourself if you wish to implement a primal barrier method or a primal-dual interior method. No matter which method you implement, your implementation should return both the primal and dual solutions.

Hand in the source code of your implementation (in machine readable format), together with a short report where you explain what method you used, derive the problem specific details of the algorithm and summarize your conclusions of numerical evaluations of your algorithm (see b) and c) below).

One application of this problem formulation is for power allocation in multicarrier (OFDM) communication systems. In this interpretation, x_i would be the power used for subcarrier i, the cost function would correspond to the total transmission rate, and the constraints $Ax \leq b$ would correspond to power constraints. For the simplest special case, $A = [1, \ldots, 1]$, i.e. a single constraint on the total power, the solution to the problem is given by the well-known water-filling algorithm, but in general it could be interesting to impose several constraints ($n > 1$) on the power levels, such as individual constraints on each x_i in order for the transmitted signal to stay below a given power mask, for example.

In Canvas, you can find both Python and Matlab skeletons, as well as some test code that compares your implementation to a reference implementation using CVX/CVXPY. Feel free to use some other programming language if you prefer.

b) Use the routine `testmaxsumlog.py/testmaxsumlog.m` (or something equivalent) to test your implementation. The files are available in Canvas. Try different problem dimensions, including $\{m = 10, n = 2\}$, $\{m = 10, n = 15\}$, $\{m = 100, n = 5\}$ and $\{m = 100, n = 20\}$. When comparing to the solution given by CVX, You should be able to obtain a relative error both in x, y and in the objective, that at least is smaller than 10^{-5}. If you get a solution that is far worse, it's most likely due to some error in your derivations or bug in your implementation, you shouldn't need to spend lots of time on tweaking parameters.

c) Comment on how the number of iterations and number of non-zero values varies with the problem dimensions.

Note that this particular cost function is one of the weak points of CVX (at least the Matlab version) when it comes to computation time, so do not draw the conclusion that CVX in general is so slow as it seems from this example.

Good luck!